

## DEPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

# JEE : Activité Pratique Etudiants

« Ingénierie Informatique : Big Data et Cloud Computing »  
II-BDCC2

Par : Assimi DIALLO

Dans ce TP nous allons créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants.

Chaque étudiant est défini par:

- Son id
- Son nom
- Son prénom
- Son email
- Sa date naissance
- Son genre : MASCULIN ou FEMININ
- Un attribut qui indique si il est en règle ou non

L'application doit offrir les fonctionnalités suivantes :

- Chercher des étudiants par nom
- Faire la pagination
- Supprimer des étudiants en utilisant la méthode
- Saisir et Ajouter des étudiants avec validation des formulaires
- Editer et mettre à jour des étudiants
- Créer une page template
- Sécuriser l'accès à l'application avec un système d'authentification basé sur Spring security en utilisant la stratégie UserDetails Service

-Classe Etudiant :

```
12  @Entity
13  @Data @AllArgsConstructor @NoArgsConstructor
14  public class Etudiant {
15      @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
16      private Long id;
17      @NotEmpty
18      @Size(min = 5, max = 20)
19      private String nom;
20      @NotEmpty
21      @Size(min = 5, max = 20)
22      private String prenom;
23      @NotEmpty
24      private String email;
25      @Temporal(TemporalType.DATE)
26      @DateTimeFormat(pattern = "yyyy-MM-dd")
27      private Date dateNaissance;
28      private Genre genre;
29      private boolean enregle;
30
31  }
```

-Genre :

```
1 package ma.enset.ap_etud.entities;
2
3 public enum Genre {
4     MASCULIN,
5     FEMININ
6 }
```

-EtudiantRepository

```
package ma.enset.ap_etud.repositories;

import ma.enset.ap_etud.entities.Etudiant;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {
    Page<Etudiant> findByNomContains(String keyword, Pageable pageable);
}
```

-Ajout d'étudiants dans la base de données :

```
@Bean
CommandLineRunner saveEtudiants(EtudiantRepository etudiantRepository){
    return args->{
        etudiantRepository.save(new Etudiant(id: null, nom: "Diallo", prenom: "Toto", email: "assim@mail.com", new Date(), Genre.MASCULIN, enregle: true));
        etudiantRepository.save(new Etudiant(id: null, nom: "Ba", prenom: "Fifi", email: "ba@mail.com", new Date(), Genre.FEMININ, enregle: true));
        etudiantRepository.save(new Etudiant(id: null, nom: "Barry", prenom: "Momo", email: "barry@mail.com", new Date(), Genre.MASCULIN, enregle: true));
        etudiantRepository.save(new Etudiant(id: null, nom: "Sow", prenom: "Nana", email: "sow@mail.com", new Date(), Genre.FEMININ, enregle: false));

        etudiantRepository.findAll().forEach(e->{
            System.out.println(e.getNom()+" "+e.getPrenom());
        });
    };
}
```

-Pour la sécurité :

-Classe AppUser :

```
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class AppRole {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long roleId;
    @Column(unique = true)
    private String roleName;
    private String description;
}
```

-Classe AppRole

```
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class AppUser {
    @Id
    private String userId;
    @Column(unique = true)
    private String username;
    private String password;
    private boolean active;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> appRoles = new ArrayList<>();
}
```

-AppUserRepository

```
package ma.enset.ap_etud.sec.repositories;

import ma.enset.ap_etud.sec.entities.AppUser;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AppUserRepository extends JpaRepository<AppUser, String> {
    AppUser findByUsername(String username);
}
```

## -AppRoleRepository

```
package ma.enset.ap_etud.sec.repositories;

import ma.enset.ap_etud.sec.entities.AppRole;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AppRoleRepository extends JpaRepository<AppRole, Long> {
    AppRole findByRoleName(String roleName);
}
```

## -SecurityService

```
1 package ma.enset.ap_etud.sec.service;
2
3
4 import ma.enset.ap_etud.sec.entities.AppRole;
5 import ma.enset.ap_etud.sec.entities.AppUser;
6
7 public interface SecurityService {
8     AppUser saveNewUser(String username, String password, String rePassword);
9     AppRole saveNewRole(String roleName, String description);
10    void addRoleToUser(String username, String roleName);
11    void removeRoleFromUser(String username, String roleName);
12    AppUser loadUserByUsername(String username);
13 }
```

## -SecurityServiceImpl

```
@Service
@Slf4j
@AllArgsConstructor
@Transactional
public class SecurityServiceImpl implements SecurityService {
    private AppUserRepository appUserRepository;
    private AppRoleRepository appRoleRepository;
    private PasswordEncoder passwordEncoder;

    @Override
    public AppUser saveNewUser(String username, String password, String rePassword) {
        if(!password.equals(rePassword)) throw new RuntimeException("Password not match");
        String hashedPWD = passwordEncoder.encode(password);
        AppUser appUser = new AppUser();
        appUser.setUserId(UUID.randomUUID().toString());
        appUser.setUsername(username);
        appUser.setPassword(hashedPWD);
        appUser.setActive(true);
        AppUser savedAppUser = appUserRepository.save(appUser);
        return savedAppUser;
    }
}
```

```

37     @Override
38     public AppRole saveNewRole(String roleName, String description) {
39         AppRole appRole = appRoleRepository.findByRoleName(roleName);
40         if(appRole!=null) throw new RuntimeException("Role "+roleName+ " already exist");
41         appRole = new AppRole();
42         appRole.setRoleName(roleName);
43         appRole.setDescription(description);
44         AppRole savedAppRole = appRoleRepository.save(appRole);
45         return savedAppRole;
46     }
47
48     @Override
49     public void addRoleToUser(String username, String roleName) {
50         AppUser appUser = appUserRepository.findByUsername(username);
51         if(appUser==null) throw new RuntimeException("User not found");
52         AppRole appRole = appRoleRepository.findByRoleName(roleName);
53         if(appRole==null) throw new RuntimeException("Role not found");
54         appUser.getAppRoles().add(appRole);
55         //appUserRepository.save(appUser); pas necessaire
56     }
57
58     @Override
59     public void removeRoleFromUser(String username, String roleName) {
60         AppUser appUser = appUserRepository.findByUsername(username);
61         if(appUser==null) throw new RuntimeException("User not found");
62         AppRole appRole = appRoleRepository.findByRoleName(roleName);
63         if(appRole==null) throw new RuntimeException("Role not found");
64         appUser.getAppRoles().remove(appRole);
65     }
66
67     @Override
68     public AppUser loadUserByUserName(String username) { return appUserRepository.findByUsername(username); }
69
70 }

```

## -UserDetailsServiceImpl

```

@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    @Autowired
    private SecurityService securityService;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        AppUser appUser = securityService.loadUserByUserName(username);

        Collection<GrantedAuthority> authorities1=
            appUser.getAppRoles() List<AppRole>
                .stream() Stream<AppRole>
                .map(role->new SimpleGrantedAuthority(role.getRoleName())) Stream<SimpleGrantedAuthority>
                .collect(Collectors.toList());

        User user = new User(appUser.getUsername(), appUser.getPassword(),authorities1);
        return user;
    }
}

```

## -SecurityConfig

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService);
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin();
        http.authorizeRequests().antMatchers("/").permitAll();
        http.authorizeRequests().antMatchers("/admin/**").hasAuthority("ADMIN");
        http.authorizeRequests().antMatchers("/user/**").hasAuthority("USER");
        http.authorizeRequests().antMatchers("/webjars/**").permitAll();
        http.authorizeRequests().anyRequest().authenticated();
        http.exceptionHandling().accessDeniedPage("/403");
    }
}
```

## -Ajout des utilisateurs et des rôles :

```
@Bean
CommandLineRunner saveUsers(SecurityService securityService){
    return args->{
        securityService.saveNewUser( username: "assimi", password: "1234", rePassword: "1234");
        securityService.saveNewUser( username: "diallo", password: "1234", rePassword: "1234");
        securityService.saveNewUser( username: "assimdll", password: "1234", rePassword: "1234");

        securityService.saveNewRole( roleName: "USER", description: "");
        securityService.saveNewRole( roleName: "ADMIN", description: "");

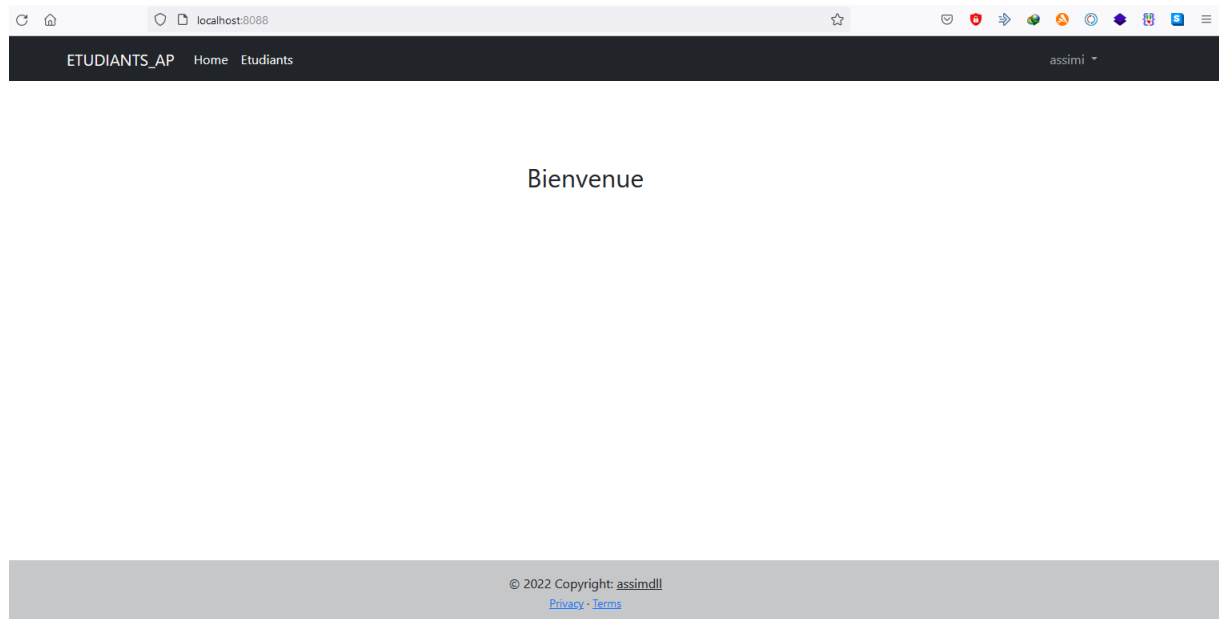
        securityService.addRoleToUser( username: "assimi", roleName: "USER");
        securityService.addRoleToUser( username: "assimi", roleName: "ADMIN");
        securityService.addRoleToUser( username: "diallo", roleName: "USER");
        securityService.addRoleToUser( username: "assimdll", roleName: "USER");
    };
}
```

On a aussi les vues avec thymeleaf :

```
templates
├── 403.html
├── editEtudiant.html
├── etudiants.html
├── etudiants2.html
├── home.html
└── template.html
```

-Présentation de l'application :

-Quand on accède sur la page home :



En cliquant sur **Etudiants** dans la navbar, on demande de se connecter pour accéder à la page :

A sign-in form with a light grey background. It features the text 'Please sign in' at the top. Below this, there are two input fields: the first contains the text 'diallo', and the second contains four dots, indicating a password field. At the bottom of the form is a blue button with the text 'Sign in'.

Je me connecte avec l'utilisateur **diallo** qui a un rôle USER :



## Gérer les étudiants

Mot  Rechercher

ID	Nom	Prenom	Email	Date	Genre	En règle
1	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
2	Baaah	Fifii	ba@mail.com	2022-04-11	MASCULIN	false
3	Barry	Momo	barry@mail.com	2022-04-11	MASCULIN	true
4	Sow	Nana	sow@mail.com	2022-04-11	FEMININ	false
5	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
7	Barry	Momo	barry@mail.com	2022-04-11	MASCULIN	true

0 1 2 3 4 5 6 7 8 9 10 11 12

Il peut voir la liste des étudiants et faire la recherche :

## Gérer les étudiants

Mot  Rechercher

ID	Nom	Prenom	Email	Date	Genre	En règle
1	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
5	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
9	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
13	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
17	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true
21	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true

0 1 2 3

Maintenant on se déconnecte pour se connecter avec un utilisateur qui a un rôle ADMIN en plus du rôle USER.

## Please sign in

You have been signed out

assimi

•••••

Sign in

ETUDIANTS\_AP

Home Etudiants

assimi

## Gérer les étudiants

Mot  Rechercher

Ajouter un étudiant

ID	Nom	Prenom	Email	Date	Genre	En règle		
1	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true	Supp.	Modif.
2	Baaah	Fifii	ba@mail.com	2022-04-11	MASCULIN	false	Supp.	Modif.
3	Barry	Momo	barry@mail.com	2022-04-11	MASCULIN	true	Supp.	Modif.
4	Sow	Nana	sow@mail.com	2022-04-11	FEMININ	false	Supp.	Modif.
5	Diallo	Toto	assim@mail.com	2022-04-11	MASCULIN	true	Supp.	Modif.
7	Barry	Momo	barry@mail.com	2022-04-11	MASCULIN	true	Supp.	Modif.

0

1

2

3

4

5

6

7

8

9

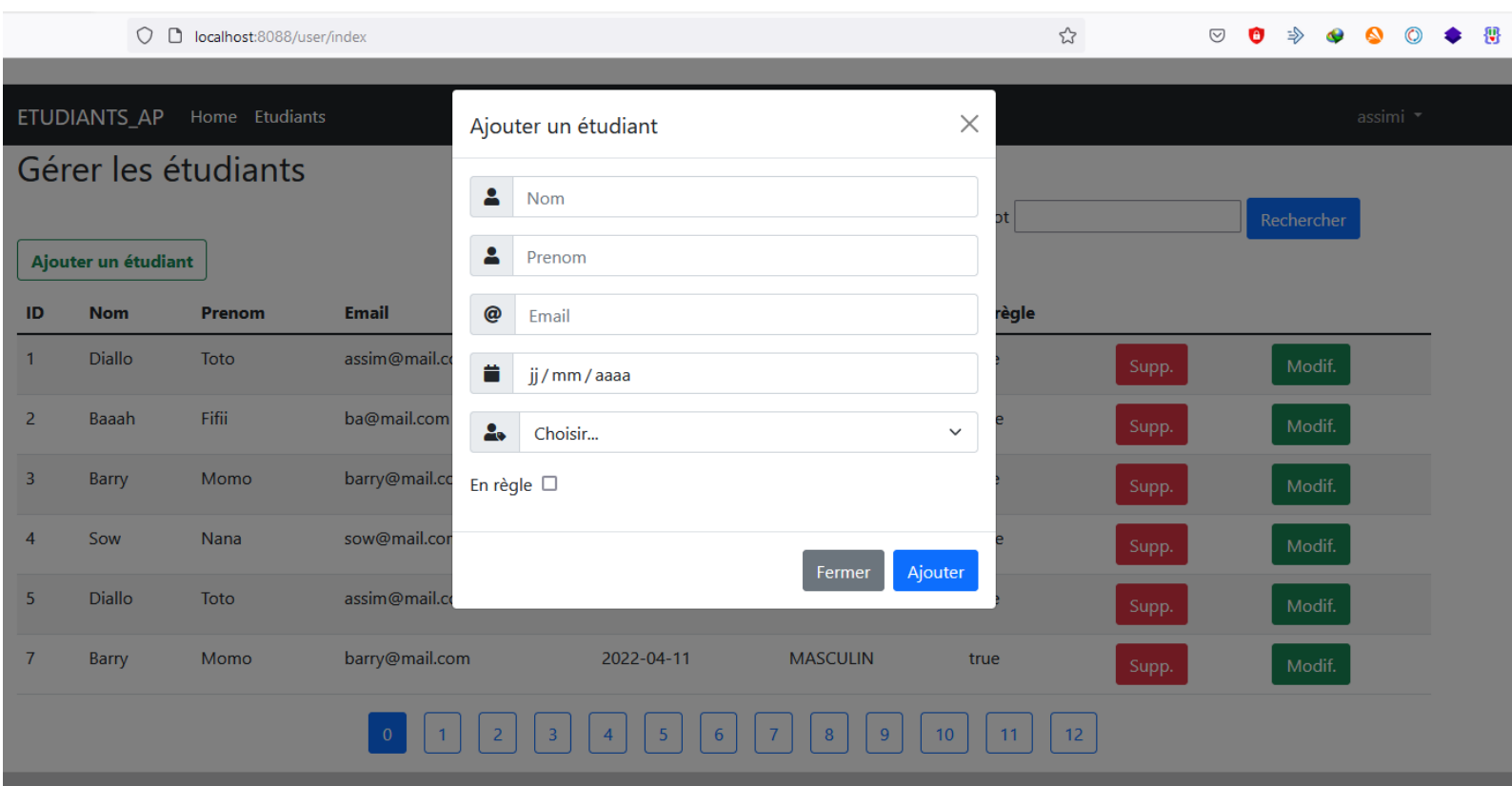
10

11

12

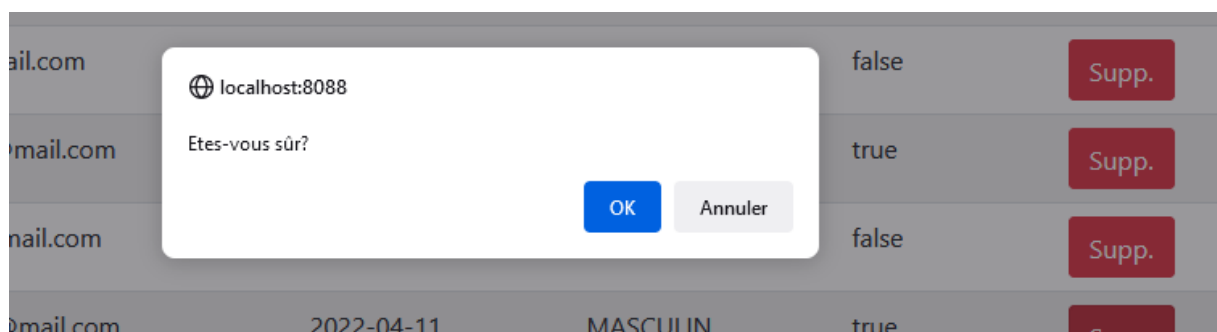
On voit donc un bouton **Ajouter un étudiant** et des boutons pour **Supprimer** et **Modifier** qui apparaissent.

Quand il clique sur le bouton **Ajouter un étudiant**



En complétant les informations et en cliquant sur le bouton **Ajouter**, l'étudiant est ajouté.

Quand il clique sur le bouton **Supp.** une confirmation pour la suppression est demandée, si on clique sur **OK**, l'étudiant est supprimé.



Quand on clique sur le bouton **Modif.** on a la possibilité de modifier l'étudiant :

The screenshot shows a web browser at the URL `localhost:8088/admin/editEtudiant?id=5&keyword=&page=0`. The page has a dark header with the text 'ETUDIANTS\_AP' and navigation links 'Home' and 'Etudiants'. A user profile 'assimi' is visible in the top right. The main content area displays a form for editing a student with ID 5. The form includes five input fields: a name field with 'Diallo', a first name field with 'Toto', an email field with 'assim@mail.com', a date field with '11 / 04 / 2022', and a dropdown menu for gender with the text 'Choisir...'. Below these fields is a checkbox labeled 'En règle' which is checked. A blue 'Modifier' button is positioned at the bottom right of the form. The footer of the page contains the copyright notice '© 2022 Copyright: [assimdl](#)' and links to 'Privacy' and 'Terms'.

ID : 5

En règle ☒

[Modifier](#)

© 2022 Copyright: [assimdl](#)  
[Privacy](#) · [Terms](#)

Pour tout ajout ou modification d'un étudiant, il y a des règles à respecter comme le fait que le nom et le prénom doivent avoir entre 5 et 20 lettres, l'adresse mail ne pas être vide...